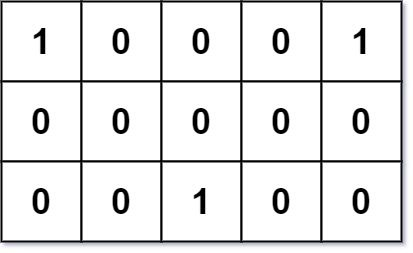
<https://leetcode.ca/2016-09-21-296-Best-Meeting-Point/>

Given an m x n binary grid grid where each 1 marks the home of one friend, return the minimal total travel distance.

The total travel distance is the sum of the distances between the houses of the friends and the meeting point.

The distance is calculated using Manhattan Distance, where distance(p1, p2) = |p2.x - p1.x| + |p2.y - p1.y|.

**Example 1:**



**Input:** grid = [[1,0,0,0,1],[0,0,0,0,0],[0,0,1,0,0]]

**Output:** 6

**Explanation:** Given three friends living at (0,0), (0,4), and (2,2).

The point (0,2) is an ideal meeting point, as the total travel distance of 2 + 2 + 2 = 6 is minimal.

So return 6.

**Example 2:**

**Input:** grid = [[1,1]]

**Output:** 1

**Constraints:**

m == grid.length

n == grid[i].length

1 <= m, n <= 200

grid[i][j] is either 0 or 1.

There will be at least two friends in the grid.

**Attempt 1: 2025-04-22**

**Solution 1: Math (60 min)**

**Refer to Deepseek**

**Problem Understanding**

The problem requires finding the optimal meeting point on a 2D grid where each '1' represents a friend's home, such that the total Manhattan distance traveled by all friends to reach this point is minimized . The Manhattan distance between two points (x1, y1) and (x2, y2) is |x1 - x2| + |y1 - y2| .

**Key Insight**

The optimal meeting point is the median of all x-coordinates and the median of all y-coordinates of the friends' homes. This is because the median minimizes the sum of absolute deviations in one dimension, and Manhattan distance allows us to treat x and y coordinates independently .

**Approach**

Collect Coordinates: Gather all row and column indices where friends are located (where grid[i][j] == 1).

Sort Coordinates: While rows are naturally collected in order, columns need explicit sorting to find the median.

Find Medians: The median of the row indices and the median of the column indices give the optimal meeting point.

Calculate Total Distance: Sum the Manhattan distances from all friends' homes to this median point .

**Solution Code**

class Solution {

public int minTotalDistance(int[][] grid) {

int m = grid.length;

int n = grid[0].length;

// Collect all row and column indices where friends are located

List<Integer> rows = new ArrayList<>();

List<Integer> cols = new ArrayList<>();

for (int i = 0; i < m; i++) {

for (int j = 0; j < n; j++) {

if (grid[i][j] == 1) {

rows.add(i);

cols.add(j);

}

}

}

// Sort the columns (rows are already in order)

Collections.sort(cols);

// Calculate median for rows and columns

int rowMedian = rows.get(rows.size() / 2);

int colMedian = cols.get(cols.size() / 2);

// Calculate total distance

int totalDistance = 0;

for (int row : rows) {

totalDistance += Math.abs(row - rowMedian);

}

for (int col : cols) {

totalDistance += Math.abs(col - colMedian);

}

return totalDistance;

}

}

**Explanation**

Coordinate Collection: We scan the grid row by row, adding the indices of all '1's to our rows and cols lists. **The rows list is automatically sorted because we process the grid in row-major order .**

Sorting Columns: Since column indices are collected in the order they appear across rows, they need to be sorted to find the median correctly .

Median Calculation: For an odd number of friends, the median is the middle element. For an even number, any point between the two middle elements gives the same minimal distance sum, so we can simply take the middle index .

Distance Calculatio: We sum the absolute differences between each friend's coordinates and the median coordinates to get the minimal total travel distance .

**Complexity Analysis**

Time Complexity: O(mn + k log k), where m and n are grid dimensions, and k is the number of friends. The dominant factor is sorting the column indices .

Space Complexity: O(k) to store the coordinates of friends .

This approach efficiently solves the problem by leveraging the properties of Manhattan distance and medians, breaking down the 2D problem into two 1D problems .

**Refer to**

[L462.Minimum Moves to Equal Array Elements II (Ref.L453)](note://WEBf26377967abd88cf360cde7f55549d27)